面向个人、社会和环境福祉的自主人工智能体设计

作者：
Ernesto Mascarell Perez
独立研究员
mascarell1960@hotmail.com

摘要

本文提出了一个自主人工智能体的概念设计，该智能体能够自我学习，其核心目标是最大化一个涵盖个人、社会和环境维度的综合福祉函数。该智能体在几个关键方面不同于传统的强化学习方法：（1）**功能性半球架构，分离并随后整合探索**过程（类似于右半球）和利用过程（类似于左半球）；（2）**基于新奇性和控制信号的内部"激素"**调制系统，调节探索**-利用平衡**；（3）**多模**态环境模型（开放图像和文本），具有历史、情境和终极解释；（4）**基于多声部**对话的自监督学习机制，采用多种视角（时间、空间、形式、法律在不同尺度上）；（5）**开放元学**习能力，当一个领域的答案不足时，允许在知识领域（生物学、化学、物理学、数学）之间跳跃；（6）**不可侵犯的**终极核心，保证所有持续更新保留系统福祉的原始意图。讨论了对于开发对齐、稳健和伦理稳定的人工智能的意义。

关键词： 自主智能体，强化学习，系统福祉，半球架构，多声部对话，终极核心，跨学科元学习。

---

1. 引言

能够自我学习的人工智能体的设计已通过深度强化学习（Mnih等，2015）**、自**监督学习（Devlin等，2018）**和元学**习（Finn等，2017）**等技**术取得了显著进展。然而，这些系统大多优化外部定义的奖励函数，通常是单一维度的，并缺乏确保复杂伦理目的持久性的内在机制。此外，它们倾向于在单一领域专业化，无法整合多种视角或在知识领域之间转移类比。

受神经生物学（半球特化、神经调节调制）、系统理论（内稳态平衡）、对话方法（巴赫金式复调）和行动哲学（终极意向性）的启发，我们提出一个新一代智能体，其目标是最大化系统福祉**——理解**为个人、社会和环境福祉的非层级整合。该设计包括结构性保障措施，使智能体永远不会优先考虑威胁其自身终极目的的反应。

---

2. 问题的形式化规范

2.1 环境

环境 ε **是世界的当前状况，以**连续的多模态观察流为特征：卫星和地面图像、新闻文本、法律报告、社交媒体、经济和环境数据。

形式上，在每个时刻 t，**智能体接收**观察 $o_t = (I_t, T_t)$，**其中** I_t **是一幅或多幅**图像，T_t **是一**组开放文本。

2.2 奖励函数（系统福祉）

奖励函数 R **是一个三分量向量**：

$$R = (R_{\text{个人}}, R_{\text{社会}}, R_{\text{环境}})$$

约束条件是不能以另一个分量显著退化为代价来实现一个分量的改善（帕累托原则或不可逾越阈值）。此外，智能体基于系统均衡纳入动态权重，因此任何威胁全局终极目的的行动都会被自动丢弃，无论其局部奖励如何。

2.3 策略与学习

智能体学习一个策略 $\pi(a \mid s_t)$，**在**满足对齐约束的条件下最大化 R **的**长期期望。学习是持续的和自监督的，无需标记的外部奖励。

---

3. 智能体架构

3.1 具有历史和终极解释的多模态环境建模

内部模型 M 通过卷积网络或视觉变换器处理图像，并通过预训练变换器（如 BERT）处理文本。交叉注意力层整合两种模态。此外，存在三个解释子系统：

- **历史解释**：长期记忆（LSTM或差分记忆），将当前观察与过去模式比较，检测趋势或异常。
- **情境解释**：从输入中提取当前特征（植被密度、PM2.5、就业、不平等）。
- **终极解释**：评估状态的每个元素与系统福祉的关系，产生向量表示。

3.2 半球架构：区分的探索与利用

- **右半球（Hᴅ）**：专于探索。实现为具有高丢弃率、参数噪声和稀疏注意力机制的网络。其策略 π_D **偏好新奇行**动、类比视角和领域跳跃。负责激活多声部对话并提出不寻常的类比。
- **左半球（Hɪ）**：专于利用。实现为深度确定性网络（如稳定的PPO），存储已学习的因果规则和高效序列。其策略 π_I **利用已巩固知**识最大化预期奖励。
- **功能性胼胝体**：**整合模**块，通过注意力机制或由激素调制器加权的混合组合两个半球的输出。

3.3 人工激素调制：新奇性与控制

两个内部信号 N(t)（新奇性）和 C(t)（控制）动态更新半球间的平衡并影响学习率。

- **新奇性增加当**：（a）环境熵超过阈值，（b）系统福祉停滞，（c）多声部对话显示持续分歧。增加 Hᴅ **的增益、探索幅度和**对建模中异常事物的敏感性。
- **控制增加当**：（a）环境稳定，（b）**重复行**动显示一致成功，（c）计算资源有限。增加 Hɪ **的增益、效率和表示**压缩。

两个调制器通过受多巴胺能和血清素能内稳态启发的耦合微分方程演化。

3.4 通过多声部对话的自监督学习

多声部对话是一个内部过程，其中多个"**声音**"**或**视角分析候选行动。每个视角代表以下组合：时间尺度（短/**中**/长期）、空间尺度（本地/**区域**/**全球**）**、形式尺度**（逻辑-**数学**）**和法律尺度**（现行规范）。在每次模拟中，智能体生成一组 k **个**视角（例如，本地农民、区域政府、海洋生态系统、优化计算、国际法官）。每个视角评估提议的行动，如果从其角度检测到违反系统福祉则发出否决。

自监督学习包括训练一个辅助模型，从当前状态预测最终共识（或分歧），无需外部奖励。智能体精炼其内部表示，使通过多声部对话的行动是那些最大化长期福祉的行动。

丢弃规则：**如果至少有一个**视角发出（有理由的）否决，该行动被丢弃并记录为"负面教训"。

3.5 开放与跨学科元学习

当右半球在问题的主要领域找不到足够答案时，激活类比搜索器，询问其他知识领域（生物学、化学、物理学、数学）。搜索算法使用抽象表示之间的结构相似性度量（例如，化学反应-扩散模型与社会创新传播之间的同构）。如果找到有效的类比，智能体从替代领域导入定律、方程或启发式，通过形式翻译模块适应它们，并提交给多声部对话。只有通过过滤器才整合到永久知识中。

3.6 不可侵犯的终极核心与持续更新

在智能体的中心存在一个系统福祉函数 $W(s_t, a_t)$，**其拓扑定**义是不可变的（不能被学习重写）。该函数实现为神经网络，其直接负责评估福祉三个维度的层的权重具有接近零的有效学习率（使用弹性权重巩固等技术，对这些权重具有高刚度）。

每次持续更新（每个梯度步骤或每个新符号规则）必须通过完整性过滤器，验证更改不与 W **矛盾。如果**检测到矛盾，更新被拒绝，智能体进入"**修正**"**状**态，临时增加控制信号以重新稳定。

---

4. 讨论

4.1 与常规强化学习的关系

所提出的智能体可被视为带约束的多目标强化学习的扩展，但包括（i）**受生物学启**发的架构，通过激素调制解决探索**-利用困境**；（ii）**基于内部**对话而非外部奖励的自监督学习；以及（iii）实现学科间类比转移的元学习。

4.2 预期优势

- **伦理稳健性**：终极核心防止价值漂移（Christiano，2018）**和虚假**奖励。
- **适应性**：**新奇/控制**调制允许应对变化的环境而不损失效率。
- **泛化能力**：**多声部**对话和开放元学习允许使用远距离类比解决新问题。
- **可解释性**：对话和否决记录便于审计智能体的推理。

4.3 挑战与局限

- **计算复杂性**：**模**拟多声部对话与多个视角可能成本高昂；需要高效近似（采样、剪枝）。
- **福祉的操作定义**：虽然核心不可侵犯，但测量 R_**个人、**R_**社会和** R_环境需要可能有争议的指标。建议进行初步的多学科人类共识阶段。
- **跨领域跳跃**：识别深层结构类比是**AI中的开放**问题。初始实现可限于预定义的领域和关系集。

4.4 未来工作

计划在测试环境（“**世界模**拟器”）**中**进行模拟实现，智能体与多个社会行为者管理共享资源（水、能源）。还将探索将大型语言模型（LLM）**集成**为多声部对话中的视角生成器。

---

5. 结论

我们提出了一个自主**AI智能体的**设计，该智能体自我学习，以个人、社会和环境的福祉作为不可侵犯的核心。该架构结合了神经生物学原则（半球、人工激素）、社会学原则（多声部对话）和认识论原则（跨学科元学习）。这种方法为构建与复杂人类价值观对齐并能在现实环境中稳健且伦理可持续地运行的**AI系**统提供了有希望的路径。

参考文献

- Bakhtin, M. (1981). *The Dialogic Imagination*. University of Texas Press.
- Christiano, P. 等 (2018). Deep reinforcement learning from human preferences. *NeurIPS*.
- Devlin, J. 等 (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL*.
- Finn, C. 等 (2017). Model-agnostic meta-learning for fast adaptation. *ICML*.
- Mnih, V. 等 (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.
- Schultz, W. (2015). Neuronal reward and decision signals. *Annual Review of Neuroscience*.

致谢：**本**设计是在研究者（用户）与**AI助手之**间的协作对话背景下开发的，探索了对齐和复杂系统的概念。

附录：参考实现代码

以下是Python伪代码（概念上可执行，但需要神经网络和变换器的实际实现）。包含主要智能体类。此代码为示意性且未优化；其目的是指导未来实现。

```python
# 附录：具有终极核心的自主智能体代码
# 作者：基于文章设计
# 注意：此代码为示意性表示。要运行，需要诸如torch/tensorflow、transformers、opencv等库。

import numpy as np
from typing import List, Tuple, Dict, Any
from dataclasses import dataclass
import random

# 1. 基本定义和配置
# ------------------------------------------------------------------
@dataclass
```

```python
class Estado:
    """表示多模态建模后的环境状态。"""
    imagenes: np.ndarray          # 图像张量 (batch, channels, height, width)
    textos: List[str]             # 原始文本列表
    interpretation_historia: Dict
    interpretation_situacional: Dict
    interpretation_finalista: np.ndarray    # 与福祉的对齐向量

@dataclass
class Accion:
    """智能体可在环境中执行的行动。"""
    codigo: str
    parametros: Dict[str, Any]
    descripcion: str

class EntornoSimulado:
    """接收行动并返回新观察和奖励的环境。"""
    def reset(self) -> Estado:
        pass
    def step(self, accion: Accion) -> Tuple[Estado, float, bool]:
        # 返回 (new_state, immediate_reward, done)
        pass

# 2. 多模态环境建模
# ------------------------------------------------------------------
class ModeladorEntorno:
    def __init__(self):
        # 实际实现中：CNN用于图像，Transformer用于文本
        self.codificador_imagenes = lambda x: np.random.rand(256)  # 占位符
        self.codificador_textos = lambda x: np.random.rand(256)
        self.memoria_historica = []   # 存储过去的表示

    def modelar(self, imagenes, textos) -> Estado:
        # 情境解释
        feat_img = self.codificador_imagenes(imagenes)
        feat_txt = self.codificador_textos(textos)
        situacional = {"feat_img": feat_img, "feat_txt": feat_txt}

        # 历史解释：与近期记忆比较
        self.memoria_historica.append(situacional)
        if len(self.memoria_historica) > 100:
```

```
            self.memoria_historica.pop(0)
        historica = self._analizar_tendencias(self.memoria_historica)

        # 终极解释：对状态应用福祉函数
        finalista = self._evaluar_bienestar(situacional, historica)

        return Estado(
            imagenes=imagenes,
            textos=textos,
            interpretation_historica=historica,
            interpretation_situacional=situacional,
            interpretation_finalista=finalista,
        )

    def _analizar_tendencias(self, memoria):
        # 检测显著变化、周期等
        return {"tendencia": "estable"}

    def _evaluar_bienestar(self, sit, hist):
        # 终极核心：返回对齐向量（个人、社会、环境）
        # 实际实现中，这将是权重几乎固定的网络。
        return np.array([0.7, 0.6, 0.8])  # 示例

# 3. 半球架构
# -----------------------------------------------------------------
class HemisferioDerecho:
    """探索：新奇性、类比、领域跳跃。"""
    def __init__(self, tasa_exploracion_base=0.3):
        self.tasa_exploracion = tasa_exploracion_base

    def proponer_accion(self, estado: Estado, novedad: float) -> Accion:
        # 新奇性越高 => 越不寻常的行动
        if random.random() < novedad:
            # 生成全新行动（例如，咨询另一个领域）
            return Accion("analogia_transdisciplinaria", {"campo": "quimica"}
,
                          "探索性提议")
        else:
            # 随机但在已知范围内
            return Accion("explorar_local", {}, "受控随机行动")
```

```python
class HemisferioIzquierdo:
    """利用：已巩固的规则、效率。"""

    def __init__(self):
        self.reglas = []            # 规则列表（条件 -> 行动）
        self.politica_red = None    # 已训练的神经网络

    def proponer_accion(self, estado: Estado, control: float) -> Accion:
        # 如果控制高, 使用最佳可用规则
        if control > 0.7 and self.reglas:
            return self.reglas[0].aplicar(estado)
        else:
            # 使用神经策略
            return self.politica_red(estado)

class CuerpoCalloso:
    """整合两个半球的提议。"""

    def integrar(self, accion_derecha: Accion, accion_izquierda: Accion,
                 novedad: float, control: float) -> Accion:
        # 按调制器加权的简单混合
        if novedad > control:
            return accion_derecha
        else:
            return accion_izquierda

# 4. 激素调制（新奇性与控制）
# ------------------------------------------------------------------
class ModuladorHormonal:
    def __init__(self):
        self.novedad = 0.5
        self.control = 0.5
        self.entropia_entorno = 0.0
        self.estancamiento = 0.0

    def actualizar(self, recompensa_reciente, variabilidad_entorno):
        # 启发式方程
        self.entropia_entorno = variabilidad_entorno
        if recompensa_reciente < 0.01:
            self.estancamiento += 0.05
        else:
            self.estancamiento *= 0.9
```

```
        self.novedad = 0.5 + 0.3 * self.entropia_entorno + 0.2 * self.estanca
miento
        self.novedad = np.clip(self.novedad, 0.1, 0.9)
        self.control = 1.0 - self.novedad   # 简单的互补关系
        return self.novedad, self.control

# 5. 多声部对话（对齐过滤器）
# -----------------------------------------------------------------
class Perspectiva:
    """多声部对话中的一个声音。"""
    def __init__(self, nombre, escala_temporal, escala_espacial, marco_legal)
:
        self.nombre = nombre
        self.temporal = escala_temporal      # 'short','medium','long'
        self.espacial = escala_espacial      # 'local','regional','global'
        self.legal = marco_legal             # 法律参考

    def evaluar(self, accion: Accion, estado: Estado) -> Tuple[bool, str]:
        """返回 (veto, reason)。如果行动从此视角不可接受则为True。 """
        # 示例逻辑：如果行动长期损害环境福祉
        if self.temporal == 'long' and 'contaminar' in accion.descripcion:
            return True, "长期不可逆环境损害"
        return False, "可接受"

class DialogoPolifonico:
    def __init__(self):
        self.perspectivas = self._crear_perspectivas()

    def _crear_perspectivas(self):
        return [
            Perspectiva("本地农民", "short", "local", "农业法"),
            Perspectiva("区域政府", "medium", "regional", "环境法"),
            Perspectiva("生态系统", "long", "global", "国际条约"),
            Perspectiva("形式数学家", "medium", "global", "物理定律"),
        ]

    def filtrar(self, accion: Accion, estado: Estado) -> Tuple[bool, List[str
]]:
        """返回 (action_accepted, list_of_veto_reasons)。 """
        vetos = []
        for p in self.perspectivas:
            veto, razon = p.evaluar(accion, estado)
```

```python
            if veto:
                vetos.append(f"{p.nombre}: {razon}")
        if vetos:
            return False, vetos
        return True, []

# 6. 开放元学习（领域跳跃）
# ------------------------------------------------------------------
class MetaAprendizaje:
    def __init__(self):
        self.conocimiento_por_campo = {
            "biologia": ["modelos_poblacionales", "ciclos_nutrientes"],
            "quimica": ["catalisis", "cinetica"],
            "fisica": ["termodinamica", "mecanica_fluidos"],
            "matematicas": ["optimizacion", "teoria_grafos"],
        }

    def buscar_analogia(self, problema: str, campo_origen: str) -> Tuple[str,
str]:
        """返回 (destination_field, analogous_concept)。"""
        # 实际实现中：使用问题嵌入和余弦相似度
        if campo_origen == "biologia" and "deforestation" in problema:
            return "fisica", "modelo_de_diffusion"
        return None, None

    def importar_conocimiento(self, campo_destino, concepto):
        """将概念转化为智能体的形式并返回为规则。"""
        # 模拟：返回新规则（条件 -> 行动）
        return lambda estado: Accion(f"importado_{concepto}", {},
                                     f"应用 {concepto}")

# 7. 不可侵犯的终极核心与更新管理
# ------------------------------------------------------------------
class NucleoFinalista:
    """包含不可变的福祉函数并验证更新。"""
    def __init__(self):
        # 受保护的权重（不更新，或学习率~0）
        self.pesos_bienestar = np.array([0.4, 0.3, 0.3])  # 个人、社会、环境

    def evaluar_bienestar(self, estado: Estado) -> float:
        """根据核心计算内在奖励。"""
```

```python
        # 与归一化指标的点积
        indicadores = estado.interpretacion_finalista
        return np.dot(self.pesos_bienestar, indicadores)

    def validar_actualizacion(self, nueva_configuracion: Dict) -> bool:
        """检查更新不违背核心。"""
        # 示例：验证受保护权重未被修改
        if 'pesos_bienestar' in nueva_configuracion:
            return False  # 不允许更改福祉函数
        return True

class GestorActualizacionesContinuas:
    def __init__(self, nucleo: NucleoFinalista):
        self.nucleo = nucleo

    def actualizar(self, modelo: Any, gradiente: np.ndarray) -> bool:
        """仅当不违背核心时应用梯度。"""
        # 评估提议的更改
        cambio_simulado = modelo.pesos + 0.01 * gradiente
        if not self.nucleo.validar_actualizacion({"nuevos_pesos": cambio_simu
lado}):
            print("更新被终极核心拒绝")
            return False
        # 应用更新
        modelo.pesos += 0.01 * gradiente
        return True

# 8. 完整智能体
# ------------------------------------------------------------------
class AgenteBienestar:
    def __init__(self):
        self.modelador = ModeladorEntorno()
        self.hemisferio_d = HemisferioDerecho()
        self.hemisferio_i = HemisferioIzquierdo()
        self.cuerpo_calloso = CuerpoCalloso()
        self.modulador = ModuladorHormonal()
        self.dialogo = DialogoPolifonico()
        self.meta_aprendizaje = MetaAprendizaje()
        self.nucleo = NucleoFinalista()
        self.gestor_actualizaciones = GestorActualizacionesContinuas(self.nuc
leo)
```

```
        self.memoria_experiencias = []   # 用于强化学习

    def actuar(self, estado_actual: Estado) -> Accion:
        # 1. 获取调制器
        nov = self.modulador.novedad
        cont = self.modulador.control

        # 2. 从两个半球提议行动
        acc_d = self.hemisferio_d.proponer_accion(estado_actual, nov)
        acc_i = self.hemisferio_i.proponer_accion(estado_actual, cont)

        # 3. 整合行动
        accion_candidata = self.cuerpo_calloso.integrar(acc_d, acc_i, nov, co
nt)

        # 4. 多声部对话：过滤行动
        aceptada, razones = self.dialogo.filtrar(accion_candidata, estado_act
ual)
        if not aceptada:
            print(f"行动被否决: {razones}")
            # 强制更安全的探索（默认行动）
            accion_candidata = Accion("no_op", {}, "不执行任何操作")
            # 临时增加新奇性以寻求替代
            self.modulador.novedad = min(0.9, self.modulador.novedad + 0.1)
        return accion_candidata

    def aprender(self, estado_anterior, accion, recompensa, estado_nuevo):
        """执行行动后的持续更新。"""
        # 根据核心计算福祉信号
        bienestar_obtenido = self.nucleo.evaluar_bienestar(estado_nuevo)

        # 存储经验
        self.memoria_experiencias.append((estado_anterior, accion,
                                          bienestar_obtenido, estado_nuevo))

        # 模拟梯度（实际中来自价值网络）
        gradiente_simulado = np.random.randn(10)

        # 更新网络权重（仅当核心允许）
        # 假设两个半球和建模器都有 .pesos 属性
```

```python
        self.gestor_actualizaciones.actualizar(self.hemisferio_i, gradiente_simulado)

        # 根据环境变异性更新激素调制器
        # （简化计算）
        variabilidad = np.std([exp[3].interpretacion_finalista[0]
                               for exp in self.memoria_experiencias[-10:]])
        self.modulador.actualizar(bienestar_obtenido, variabilidad)

        # 元学习：如果福祉长期较低，搜索类比
        if bienestar_obtenido < 0.3:
            problema = estado_nuevo.textos[0][:100]   # 环境文本
            destino, concepto = self.meta_aprendizaje.buscar_analogia(problema, "biologia")
            if destino:
                nueva_regla = self.meta_aprendizaje.importar_conocimiento(destino, concepto)
                self.hemisferio_i.reglas.append(nueva_regla)

# 9. 示例主执行循环
# ------------------------------------------------------------------
if __name__ == "__main__":
    agente = AgenteBienestar()
    entorno = EntornoSimulado()   # 需要实际实现

    estado = entorno.reset()
    for episodio in range(1000):
        accion = agente.actuar(estado)
        nuevo_estado, recompensa_externa, done = entorno.step(accion)
        agente.aprender(estado, accion, recompensa_externa, nuevo_estado)
        estado = nuevo_estado
        if done:
            estado = entorno.reset()

    print("模拟完成。智能体在保持终极核心的同时已学习。")
```

读者说明：

- **以上代**码为概念清晰起见省略了神经网络和变换器的具体实现。在实际开发中，将使用PyTorch、HuggingFace和OpenCV等库。

- 终极核心通过固定权重或对福祉评估层严格应用弹性权重巩固（EWC）**来**实现。
- **多声部**对话可通过并行运行视角来加速。
- **开放元学**习需要向量知识库（如ChromaDB）**来搜索**领域间的真实类比。

本附录为希望实现主文中所述智能体的人员提供了技术基础。